
Boot Camp - Bank

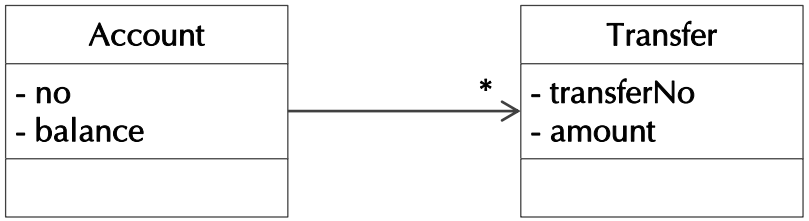
1. 도메인 모델

posco

1.1 도메인 모델

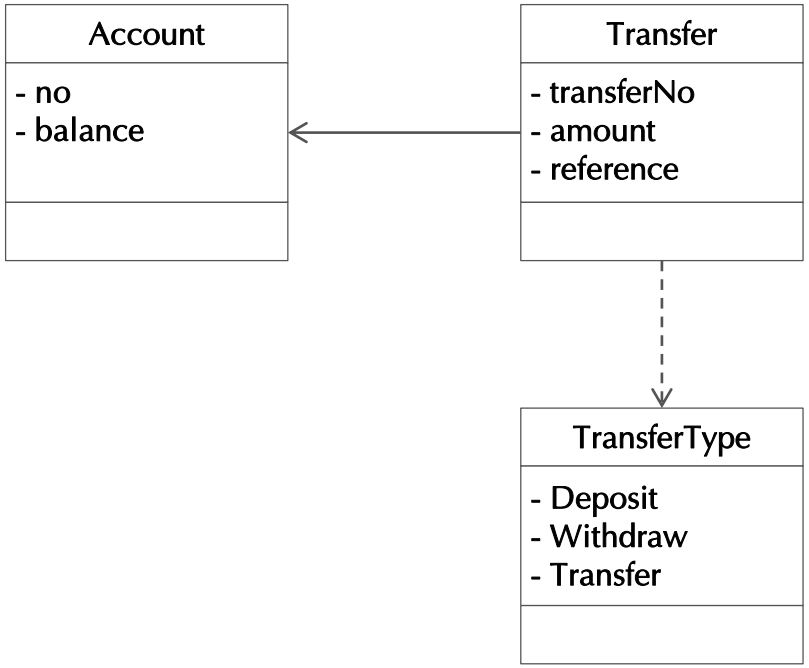
✓ 은행의 주요 업무는 계좌에 대한 입금, 출금, 계좌이체입니다.

✓ 입금, 출금, 계좌이체는 모두 이력을 기록해야 하고 계좌별로 이력을 확인 할 수 있어야 합니다.



1.2 설계 모델

- ✓ Bank 서비스는 Account, Transfer 두 개의 Entity를 가집니다.
- ✓ AccountService는 계좌 개설, 입금, 출금 기능을 제공합니다.
- ✓ TransferService는 계좌간 이체 기능을 제공합니다.
- ✓ AccountStore는 계좌를 관리하고 TransferStore는 입/출금, 계좌 이체 이력을 관리합니다.
- ✓ 계좌 이체 중 한 계좌에서 오류가 발생하면 트랜잭션을 이용하여 두 계좌 모두 잔액의 변경이 없어야 합니다.



1.3 구현 내용

✓bank-domain, bank-service, bank-store-jpa를 구현합니다.

✓기능별 URL은 아래와 같으며 순차적으로 실행했을 때 200 OK 응답을 반환해야 합니다.

기능	URL
계좌개설	POST http://localhost:8080
입금	PUT http://localhost:8080/{accountNo}/deposit/100
출금	PUT http://localhost:8080/{accountNo}/withdraw/20
계좌조회	GET http://localhost:8080/{accountNo}
계좌이체	POST http://localhost:8080/{fromAccountNo}/{toAccountNo}/30
거래내역	GET http://localhost:8080/{accountNo}/transfer

1.4 Service간 DI

✓TransferService에서 AccountService으로의 참조는 Service와 Lifecycle간 순환 참조를 발생시킵니다.

✓따라서 TransferService에서 AccountService의 참조는 Lifecycle에서 직접 Injection 시켜줍니다.

TransferService.java

```
public interface TransferService {
    public void setAccountService(AccountService accountService);
    public void transfer(String fromAccountNo
                         , String toAccountNo
                         , long amount);
}
```

ServiceLifecycler.java

```
@Component
public class ServiceLifecycler implements ServiceLifecycle {
    //
    private final AccountService accountService;
    private final TransferService transferService;

    public ServiceLifecycler(AccountService accountService
                             , TransferService transferService) {
        this.accountService = accountService;
        this.transferService = transferService;

        // manual dependency injection
        this.transferService.setAccountService(this.accountService);
    }
```